The doBy package

true

Abstract

The doBy is one of several general utility packages on CRAN. An abstract of less than 150 words.

Introduction

The doBy package [@doby] appeared on CRAN [@CRAN] in 2006 and, much to our surprise, the package is still being used. The package originally grew out of a need to calculate groupwise summary statistics (much in the spirit of PROC SUMMARY of the SAS system, [@procsummary]). The name comes from doing som computations when data is stratified by the value of some variables. Today the package contains many different utilities. In this paper we focus 1) on these "doing by" functions, 2) on functions related to linears estimates and contrasts and 3) on some of the miscellaneous functions in the package.

Functions related to groupwise computations

A working dataset

The CO2 data frame comes from an experiment on the cold tolerance of the grass species *Echinochloa crus-galli*. To limit the amount of output we modify names and levels of variables as follows

```
data(CO2)
CO2 <- within(CO2, {
    Treat <- Treatment
    Treatment <- NULL
    levels(Treat) <- c("nchil","chil")
    levels(Type)  <- c("Que","Mis")
})
CO2 <- subset(CO2, Plant %in% c("Qn1", "Qc1", "Mn1", "Mc1"))
CO2
```

```
##    Plant Type conc uptake Treat
## 1    Qn1  Que   95   16.0 nchil
## 2    Qn1  Que  175   30.4 nchil
## 3    Qn1  Que  250   34.8 nchil
## 4    Qn1  Que  350   37.2 nchil
## 5    Qn1  Que  500   35.3 nchil
## 6    Qn1  Que  675   39.2 nchil
## 7    Qn1  Que 1000   39.7 nchil
## 22   Qc1  Que   95   14.2  chil
## 23   Qc1  Que  175   24.1  chil
## 24   Qc1  Que  250   30.3  chil
## 25   Qc1  Que  350   34.6  chil
## 26   Qc1  Que  500   32.5  chil
## 27   Qc1  Que  675   35.4  chil
```

```
## 28   Qc1  Que 1000   38.7  chil
## 43   Mn1  Mis   95   10.6 nchil
## 44   Mn1  Mis  175   19.2 nchil
## 45   Mn1  Mis  250   26.2 nchil
## 46   Mn1  Mis  350   30.0 nchil
## 47   Mn1  Mis  500   30.9 nchil
## 48   Mn1  Mis  675   32.4 nchil
## 49   Mn1  Mis 1000   35.5 nchil
## 64   Mc1  Mis   95   10.5  chil
## 65   Mc1  Mis  175   14.9  chil
## 66   Mc1  Mis  250   18.1  chil
## 67   Mc1  Mis  350   18.9  chil
## 68   Mc1  Mis  500   19.5  chil
## 69   Mc1  Mis  675   22.2  chil
## 70   Mc1  Mis 1000   21.9  chil
```

The summaryBy function

The summaryBy function is used for calculating quantities like **the mean and variance of numerical variables x and y for each combination of two factors A and B$**. Notice: A functionality similar to summaryBy is provided by aggregate from base R. [SH: DUMT at variabel skal navngives; det skal den ikke i aggregate].

```
myfun1 <- function(x){c(m=mean(x), s=sd(x))}
summaryBy(cbind(conc, uptake, lu=log(uptake)) ~ Plant,
          data=CO2, FUN=myfun1)
```

```
##   Plant conc.m conc.s uptake.m uptake.s  lu.m   lu.s
## 1   Qn1    435  317.7    33.23    8.215 3.467 0.3189
## 2   Qc1    435  317.7    29.97    8.335 3.356 0.3446
## 3   Mn1    435  317.7    26.40    8.694 3.209 0.4234
## 4   Mc1    435  317.7    18.00    4.119 2.864 0.2622
```

Instead of formula we may specify a list containing the left hand side and the right hand side of a formula (This is a feature of summaryBy and it does not work with aggregate.).

```
## FIXME: Will fail because of log(uptake)
summaryBy(list(c("conc", "uptake", "lu=log(uptake)"), "Plant"), data=CO2, FUN=myfun1)
```

```
##   Plant conc.m conc.s uptake.m uptake.s  lu.m   lu.s
## 1   Qn1    435  317.7    33.23    8.215 3.467 0.3189
## 2   Qc1    435  317.7    29.97    8.335 3.356 0.3446
## 3   Mn1    435  317.7    26.40    8.694 3.209 0.4234
## 4   Mc1    435  317.7    18.00    4.119 2.864 0.2622
```

Various convenient abbreviations are available, e.g. the following, where left hand side dot refers to "all numeric variables" while the right hand side dot refers to "all factor variables".

```
summaryBy(.~., data=CO2, FUN=myfun1)
```

```
##   Plant Type Treat conc.m conc.s uptake.m uptake.s
## 1   Qn1  Que nchil    435  317.7    33.23    8.215
## 2   Qc1  Que  chil    435  317.7    29.97    8.335
## 3   Mn1  Mis nchil    435  317.7    26.40    8.694
## 4   Mc1  Mis  chil    435  317.7    18.00    4.119
```

```
## same as summaryBy(list(c("."), c(".")), data=CO2, FUN=myfun1)
```

The orderBy function

Ordering (or sorting) a data frame is possible with the `orderBy` function. Suppose we want to order the rows of the the `CO2` data by increasing values of `conc` and decreasing value of `uptake` (within `code`):

```
x1 <- orderBy(~ conc - uptake, data=CO2)
head(x1)
```

```
##    Plant Type conc uptake Treat
## 1    Qn1  Que   95   16.0 nchil
## 22   Qc1  Que   95   14.2  chil
## 43   Mn1  Mis   95   10.6 nchil
## 64   Mc1  Mis   95   10.5  chil
## 2    Qn1  Que  175   30.4 nchil
## 23   Qc1  Que  175   24.1  chil
```

An equivalent form is:

```
orderBy(c("conc", "-uptake"), data=CO2) %>% head
```

```
##    Plant Type conc uptake Treat
## 64   Mc1  Mis   95   10.5  chil
## 43   Mn1  Mis   95   10.6 nchil
## 22   Qc1  Que   95   14.2  chil
## 1    Qn1  Que   95   16.0 nchil
## 65   Mc1  Mis  175   14.9  chil
## 44   Mn1  Mis  175   19.2 nchil
```

The splitBy function

Suppose we want to split `CO2` into a list of dataframes:

```
x1 <- splitBy(~ Plant + Type, data=CO2)
x1
```

```
##   listentry Plant Type
## 1   Qn1|Que   Qn1  Que
## 2   Qc1|Que   Qc1  Que
## 3   Mn1|Mis   Mn1  Mis
## 4   Mc1|Mis   Mc1  Mis
```

The result is a list:

```
lapply(x1, head, 2)
```

```
## $`Qn1|Que`
##   Plant Type conc uptake Treat
## 1   Qn1  Que   95   16.0 nchil
## 2   Qn1  Que  175   30.4 nchil
##
## $`Qc1|Que`
##    Plant Type conc uptake Treat
## 22   Qc1  Que   95   14.2  chil
## 23   Qc1  Que  175   24.1  chil
```

```
##
## $`Mn1|Mis`
##    Plant Type conc uptake Treat
## 43   Mn1  Mis   95   10.6 nchil
## 44   Mn1  Mis  175   19.2 nchil
##
## $`Mc1|Mis`
##    Plant Type conc uptake Treat
## 64   Mc1  Mis   95   10.5  chil
## 65   Mc1  Mis  175   14.9  chil
```

The subsetBy function

Suppose we want to select those rows within each treatment for which the uptake is larger than 75% quantial of uptake (within the treatment). This is achieved by:

```
x2 <- subsetBy(~Treat, subset=uptake > quantile(uptake, prob=0.75), data=CO2)
x2
```

```
##          Plant Type conc uptake Treat
## nchil.4    Qn1  Que  350   37.2 nchil
## nchil.6    Qn1  Que  675   39.2 nchil
## nchil.7    Qn1  Que 1000   39.7 nchil
## nchil.49   Mn1  Mis 1000   35.5 nchil
## chil.25    Qc1  Que  350   34.6  chil
## chil.26    Qc1  Que  500   32.5  chil
## chil.27    Qc1  Que  675   35.4  chil
## chil.28    Qc1  Que 1000   38.7  chil
```

The transformBy function

The `transformBy` function is analogous to the `transform` function except that it works within groups. For example:

```
x <- transformBy(~Treat, data=CO2,
                 minU=min(uptake), maxU=max(uptake),
                 range = diff(range(uptake)))
head(x)
```

```
##         Plant Type conc uptake Treat minU maxU range
## nchil.1   Qn1  Que   95   16.0 nchil 10.6 39.7  29.1
## nchil.2   Qn1  Que  175   30.4 nchil 10.6 39.7  29.1
## nchil.3   Qn1  Que  250   34.8 nchil 10.6 39.7  29.1
## nchil.4   Qn1  Que  350   37.2 nchil 10.6 39.7  29.1
## nchil.5   Qn1  Que  500   35.3 nchil 10.6 39.7  29.1
## nchil.6   Qn1  Que  675   39.2 nchil 10.6 39.7  29.1
```

The lmBy function

```
m <- lmBy(uptake ~ conc | Treat, data=CO2)
lapply(m, function(z) coef(summary(z)))
```

```
## $nchil
```

```
##             Estimate Std. Error t value  Pr(>|t|)
## (Intercept) 20.82342   3.092430   6.734 2.092e-05
## conc         0.02067   0.005889   3.510 4.304e-03
##
## $chil
##             Estimate Std. Error t value  Pr(>|t|)
## (Intercept) 17.01814   3.668315   4.639 0.0005709
## conc         0.01602   0.006986   2.293 0.0407168
```

Functions related linear estimates and contrasts

A linear function of a p–dimensional parameter vector β has the form

$$C = L\beta$$

where L is a $q \times p$ matrix which we call the `Linear Estimate Matrix` or simply `LE-matrix`. The corresponding linear estimate is $\hat{C} = L\hat{\beta}$. A linear hypothesis has the form $H_0 : L\beta = m$ for some q dimensional vector m.

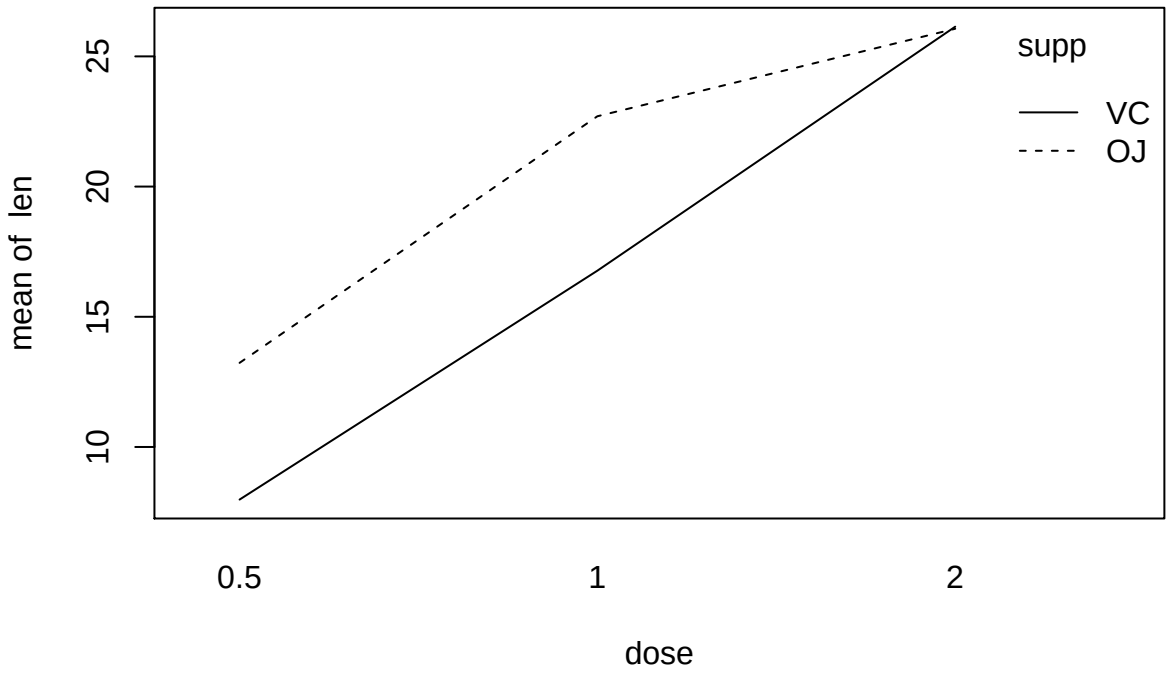
A working dataset

The response is the length of odontoblasts cells (cells responsible for tooth growth) in 60 guinea pigs. Each animal received one of three dose levels of vitamin C (0.5, 1, and 2 mg/day) by one of two delivery methods, (orange juice (coded as OJ) or ascorbic acid (a form of vitamin C and (coded as VC)).

```
ToothGrowth$dose <- factor(ToothGrowth$dose)
head(ToothGrowth, 4)
```

```
##    len supp dose
## 1  4.2   VC  0.5
## 2 11.5   VC  0.5
## 3  7.3   VC  0.5
## 4  5.8   VC  0.5
```

```
with(ToothGrowth, interaction.plot(dose, supp, len))
```



Computing linear estimates

Focus on additive model:

```
tooth1 <- lm(len ~ dose + supp, data=ToothGrowth)
tooth2 <- lm(len ~ dose * supp, data=ToothGrowth)
anova(tooth1, tooth2)
```

```
## Analysis of Variance Table
##
## Model 1: len ~ dose + supp
## Model 2: len ~ dose * supp
##   Res.Df RSS Df Sum of Sq    F Pr(>F)
## 1     56 820
## 2     54 712  2       108 4.11  0.022 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Consider computing the estimated length for each dose of orange juice (OJ): One option: Construct the LE–matrix L directly and then invoke `linest`:

```
L <- matrix(c(1, 0, 0, 0,
              1, 1, 0, 0,
              1, 0, 1, 0), nrow=3, byrow=T)
c1 <- linest(tooth1, L)
c1
```

```
## Coefficients:
##      estimate std.error statistic     df p.value
## [1,]   12.455     0.988    12.603 56.000       0
## [2,]   21.585     0.988    21.841 56.000       0
## [3,]   27.950     0.988    28.281 56.000       0
```

We can do:

```
summary(c1)
```

```
## Coefficients:
##      estimate std.error statistic     df p.value
## [1,]   12.455     0.988    12.603 56.000       0
## [2,]   21.585     0.988    21.841 56.000       0
## [3,]   27.950     0.988    28.281 56.000       0
##
## Grid:
## NULL
##
## L:
##      [,1] [,2] [,3] [,4]
## [1,]    1    0    0    0
## [2,]    1    1    0    0
## [3,]    1    0    1    0
```

```
coef(c1)
```

```
##   estimate std.error statistic df   p.value
## 1    12.45    0.9883     12.60 56 5.490e-18
## 2    21.58    0.9883     21.84 56 4.461e-29
## 3    27.95    0.9883     28.28 56 7.627e-35
```

```
confint(c1)
```

```
##   0.025 0.975
## 1 10.48 14.43
## 2 19.61 23.56
## 3 25.97 29.93
```

The matrix L can be generated as follows:

```
L <- LE_matrix(tooth1, effect="dose", at=list(supp="OJ"))
L
```

```
##      (Intercept) dose1 dose2 suppVC
## [1,]           1     0     0      0
## [2,]           1     1     0      0
## [3,]           1     0     1      0
```

There are various alternatives:

```
c1 <- esticon(tooth1, L)
c1
```

```
##      estimate std.error statistic p.value  beta0 df
## [1,]   12.455     0.988    12.603   0.000  0.000 56
## [2,]   21.585     0.988    21.841   0.000  0.000 56
## [3,]   27.950     0.988    28.281   0.000  0.000 56
```

Notice: `esticon` has been in the `doBy` package for many years; `linest` is a newer addition. Yet another alternative in this case is to generate a new data frame and then invoke predict (but this approach is not generally applicable, see later):

```
nd <- data.frame(dose=c('0.5', '1', '2'), supp='OJ')
nd
```

```
##   dose supp
## 1  0.5   OJ
```

```
## 2    1   OJ
## 3    2   OJ
```

```
predict(tooth1, newdata=nd)
```

```
##     1     2     3
## 12.45 21.58 27.95
```

Least-squares means (LS–means)

A related question could be: What is the estimated length for each dose if we ignore the source of vitamin C (i.e. whether it is OJ or VC). One approach would be to fit a model in which source does not appear:

```
tooth0 <- update(tooth1, . ~ . - supp)
L0 <- LE_matrix(tooth0, effect="dose")
L0
```

```
##      (Intercept) dose1 dose2
## [1,]           1     0     0
## [2,]           1     1     0
## [3,]           1     0     1
```

```
linest(tooth0, L=L0)
```

```
## Coefficients:
##      estimate std.error statistic     df p.value
## [1,]   10.605     0.949    11.180 57.000       0
## [2,]   19.735     0.949    20.805 57.000       0
## [3,]   26.100     0.949    27.515 57.000       0
```

« »= @

An alternative would be to stick to the original model but compute the estimate for an "average vitamin C source". That would correspond to giving weight 1/2 to each of the two vitamin C source parameters. However, as one of the parameters is already set to zero to obtain identifiability, we obtain the LE–matrix L as

```
L1 <- matrix(c(1, 0, 0, 0.5,
               1, 1, 0, 0.5,
               1, 0, 1, 0.5), nrow=3, byrow=T)
linest(tooth1, L=L1)
```

```
## Coefficients:
##      estimate std.error statistic     df p.value
## [1,]   10.605     0.856    12.391 56.000       0
## [2,]   19.735     0.856    23.058 56.000       0
## [3,]   26.100     0.856    30.495 56.000       0
```

Such a particular linear estimate is sometimes called a least-squares mean or an LSmean or a marginal mean. Notice that the parameter estimates under the two approaches are identical. This is is because data is balanced: There are 10 observations per supplementation type. Had data not been balanced, the estimates would in general have been different.

Notice: One may generate L automatically with

```
L1 <- LE_matrix(tooth1, effect="dose")
L1
```

```
##      (Intercept) dose1 dose2 suppVC
```

```
## [1,]           1     0     0    0.5
## [2,]           1     1     0    0.5
## [3,]           1     0     1    0.5
```

Notice: One may obtain the LSmean directly as:

```
LSmeans(tooth1, effect="dose")
```

```
## Coefficients:
##      estimate std.error statistic     df p.value
## [1,]   10.605     0.856    12.391 56.000       0
## [2,]   19.735     0.856    23.058 56.000       0
## [3,]   26.100     0.856    30.495 56.000       0
```

%%

Additive model

%%

%% Returning to the `ToothGrowth` data, orange juice and ascorbic %% acid are just two of many ways of supplying vitamin C (citrus and lime %% juice would be two alternatives). Here one can therefore argue, that %% it would make sense to estimate the the effect for each dose for an %% "average vitamin C source":

%% « »= %% LSmeans(tooth1, effect="dose") %% @

which is the same as

```
L <- LE_matrix(tooth1, effect="dose")
le <- linest(tooth1, L=L)
coef(le)
```

```
##   estimate std.error statistic df   p.value
## 1    10.60    0.8559     12.39 56 1.109e-17
## 2    19.73    0.8559     23.06 56 2.885e-30
## 3    26.10    0.8559     30.50 56 1.444e-36
```

%%

Interaction model

%%

For a model with interactions, the LSmeans are

```
LSmeans(tooth2, effect="dose")
```

```
## Coefficients:
##      estimate std.error statistic     df p.value
## [1,]   10.605     0.812    13.060 54.000       0
## [2,]   19.735     0.812    24.304 54.000       0
## [3,]   26.100     0.812    32.143 54.000       0
```

In this case, the LE–matrix is

```
L <- LE_matrix(tooth2, effect="dose")
t(L)
```

```
##              [,1] [,2] [,3]
## (Intercept)   1.0  1.0  1.0
## dose1         0.0  1.0  0.0
## dose2         0.0  0.0  1.0
## suppVC        0.5  0.5  0.5
## dose1:suppVC  0.0  0.5  0.0
## dose2:suppVC  0.0  0.0  0.5
```

Using the at= argument

Consider random regression model:

```
library(lme4)
chick <- lmer(weight ~ Time * Diet + (0 + Time | Chick),
           data=ChickWeight)
coef(summary(chick))
```

```
##             Estimate Std. Error t value
## (Intercept)   33.218     1.7697 18.7701
## Time           6.339     0.6103 10.3855
## Diet2         -4.585     3.0047 -1.5258
## Diet3        -14.968     3.0047 -4.9815
## Diet4         -1.454     3.0177 -0.4818
## Time:Diet2     2.271     1.0367  2.1902
## Time:Diet3     5.084     1.0367  4.9043
## Time:Diet4     3.217     1.0377  3.1004
```

Using (transformed) covariates

Consider the following subset of the CO2 dataset:

```
data(CO2)
CO2 <- transform(CO2, Treat=Treatment, Treatment=NULL)
levels(CO2$Treat) <- c("nchil","chil")
levels(CO2$Type) <- c("Que","Mis")
ftable(xtabs( ~ Plant + Type + Treat, data=CO2), col.vars=2:3)
```

```
##       Type    Que        Mis
##       Treat nchil chil nchil chil
## Plant
## Qn1             7    0     0    0
## Qn2             7    0     0    0
## Qn3             7    0     0    0
## Qc1             0    7     0    0
## Qc3             0    7     0    0
## Qc2             0    7     0    0
## Mn3             0    0     7    0
## Mn2             0    0     7    0
## Mn1             0    0     7    0
## Mc2             0    0     0    7
## Mc3             0    0     0    7
## Mc1             0    0     0    7
```

Below, the covariate conc is fixed at the average value:

```
co2.lm1 <- lm(uptake ~ conc + Type + Treat, data=CO2)
LSmeans(co2.lm1, effect="Treat")
```

```
## Coefficients:
##      estimate std.error statistic     df p.value
## [1,]   30.643     0.956    32.066 80.000       0
## [2,]   23.783     0.956    24.888 80.000       0
```

If we use `log(conc)` instead we will get an error when calculating LS–means:

```
co2.lm <- lm(uptake ~ log(conc) + Type + Treat, data=CO2)
LSmeans(co2.lm, effect="Treat")
```

In this case one can do

```
co2.lm2 <- lm(uptake ~ log.conc + Type + Treat,
              data=transform(CO2, log.conc=log(conc)))
LSmeans(co2.lm2, effect="Treat")
```

```
## Coefficients:
##      estimate std.error statistic     df p.value
## [1,]   30.643     0.761    40.261 80.000       0
## [2,]   23.783     0.761    31.248 80.000       0
```

This also highlights what is computed: The average of the log of `conc`; not the log of the average of `conc`. In a similar spirit consider

```
co2.lm3 <- lm(uptake ~ conc + I(conc^2) + Type + Treat, data=CO2)
LSmeans(co2.lm3, effect="Treat")
```

```
## Coefficients:
##      estimate std.error statistic     df p.value
## [1,]   34.543     0.982    35.191 79.000       0
## [2,]   27.683     0.982    28.202 79.000       0
```

Above `I(conc^2)` is the average of the squared values of `conc`; not the square of the average of `conc`, cfr. the following.

```
co2.lm4 <- lm(uptake ~ conc + conc2 + Type + Treat, data=
              transform(CO2, conc2=conc^2))
LSmeans(co2.lm4, effect="Treat")
```

```
## Coefficients:
##      estimate std.error statistic     df p.value
## [1,]   30.643     0.776    39.465 79.000       0
## [2,]   23.783     0.776    30.630 79.000       0
```

If we want to evaluate the LS–means at `conc=10` then we can do:

```
LSmeans(co2.lm4, effect="Treat", at=list(conc=10, conc2=100))
```

```
## Coefficients:
##      estimate std.error statistic    df p.value
## [1,]    14.74      1.70      8.66 79.00       0
## [2,]     7.88      1.70      4.63 79.00       0
```

Acknowledgements

Credit is due to Dennis Chabot, Gabor Grothendieck, Paul Murrell, Jim Robison-Cox and Erik Jørgensen for reporting various bugs and making various suggestions to the functionality in the `doBy` package.

Summary

This file is only a basic article template. For full details of *The R Journal* style and information on how to prepare your article for submission, see the Instructions for Authors.